JACG NEWSLETTER

Vol.4 No.8

Apr. 1985

Single Copy Price $1.50

THE JERSEY ATARI COMPUTER GROUP

884-1642 JACG HOTLINE 884-1642

From the Editor's Desk...

This month's word is CAVEAT. From the Latin: let him beware; a warning. In the January issue we published a letter from JACG member William Hough concerning his unfortunate luck with two Astra disk drives. We have since received a packet of materials from Astra Systems, Inc. letting us know of their distress at our printing of "Mr. Hough's mendacious assertions." The letter, from Astra plant manager Drew Featherston, takes us to task for publishing such a "libelous and defamatory letter" and chastises us for not verifying the facts and allowing Astra to present their case.

In the second case we are most guilty. In the fairness of journalism it was, indeed, a professional discourtesy not to invite Astra to counter Mr. Hough's charges. For that we apologize. Being a small publication intended for distribution to our membership and some other user groups we never have been so egocentric to think of ourselves in the same league as ANTIC or ANALOG magazines. Mr. Featherston, however, points out that these publications refused to print Mr. Hough's "malicious missive" because they "recognized their legal obligations."

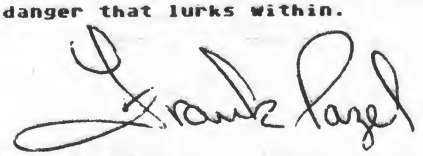
A review of the materials from Astra makes it apparent that there was considerable mix-up in communications between Astra and Mr. Hough. Astra apparently tried to set things straight but the corporate-customer relationship had too far soured. In all fairness, it further seems that Astra has cured the basic problem of excessive overheating. Copies of letters from satisfied customers indicate an excellent relationship with Astra.

In the final analysis it seems what we have here is the sort of unfortunate situation described by Tom Peters in his popular book, "In Search of Excellence", wherein the true human sensitivity of a corporation can get lost in administrative procedures. This is not to say either side in this dispute was/is right or wrong. The perceptions are what count.

As the official organ of our group we feel it is our responsibility to report on Atari-relative matters. That is why we published the letter originally. We had received many reports of Astra drives failing. We are sure that Mr. Featherston (correctly) feels it his right and responsibility to protect the good name of Astra Systems, Inc. Again, in fairness, it seems that Astra is doing everything they can to rectify these past problems and are now producing an effective and dependable product. We would invite you to look through the dossier of materials Mr. Featherston sent and inform yourself. See any club officer for a copy of these materials.

In the future we certainly will make it a point to continue to print both positive and negative criticism of products which our membership might come in contact with. We will also make it a point to invite differing and opposing views. At the risk of appearing sloppily sentimental, we think that is the American way.

Briefly, another caveat should be observed as you peruse this colorful issue. It is April, after all, and those who have been among us more than a year know the danger that lurks within.



Frank Pazel
Editor-in-Chief, JACG Newsletter

INSIDE
22 ARTICLES
Table of Contents - Page 3

MARK YOUR CALENDARS!!

JACG Meeting Schedule

May 11, 1985
June 8, 1985
July 13, 1985
August 10, 1985

From The Conn.....

Bunch of topics this month..... April, ah yes, April. The foolish month or time for pranks. Will April 1st see the likes of two (or more) new Atari computers? Will April 30th see two new Atari computers? According to Atari, who does little talking except for heavy boasting for the past several months, the XE and ST computers will make their debut this month.

Yes, that's right. Apparently, the 130XE computer will soon appear at a mass merchandiser near you. The 65XE is unlikely to be produced in this man's opinion. Why? Because the difference in price between the 65XE and the 130XE is reported to be only $25 or so. Therefore, who would not spend the extra bucks for the machine with twice the memory (128K vs. 64K). It's the old 600XL/800XL marketing deal all over again. The 600XL computer with its limited 16K of memory never sold in enough quantities to make it profitable because the 800XL was so close to its price.

What about the 130ST/520ST you ask? A similar deal here as well. If you were Atari, make that, if you were Jack Tramiel (sounds better doesn't it?), and you wanted to make a splash with your new JackIntosh, which machine would you bring out first? That's right, the 520ST. Nobody else has got anything like it for anywhere near the price.

Apple has the Mac, a 128K machine with black and white graphics at only $1800. Of course, you could spend another $1000 (that's 1000 big ones) and have a 512K Mac. By Atari first coming out with the 512K 520ST, for only $600, they have quartered the price of their competitions machine. But will Atari do it this month? Will Bounty Bob get his man? Will JR and Sue-Ellen get back in the sack? Beats me!

Atari is not with out its faults. To be sure! Big Jack promised support of software authors at the January CES software authors meeting. He said he would provide technical, financial and any other type of support he could. Within two months, it was announced by Atari that any software developer who wanted a machine would have to spend $5000 for a development ST computer. The first number was $2500 announced in late January. Whahappnd? So much for JTari's support of software developers.

There is a new User Group Coordinator at Atari. His name is David Duberman. Some of you will remember him from his technical days at Antic. I talked with Dave recently about what a User Group Coordinator does. He said, don't know. I asked what Atari plans to do for User Groups and he said, big things. I said, like what? He said, can't tell. I said when will you know, he said, don't know. Brian Kerr, another Atari person who I have been talking with since last fall has been telling me the same thing. Atari plans big things for User Groups. Well, April 1st is just around the corner and I wonder if these Atari folks are practicing for the first. I mean, here it is, a few moments (supposedly) before the introduction of the new computers, and Atari still doesn't know what it is doing in the area of User Group support. Maybe we should get Sam, Leonard, Jack or any of the other boys out here for one of our meetings and find out what the hell is going on.

New topic: Broderbund. I want to thank Bill Holt for visiting our meeting last month and demoing Print Shop and other Broderbund titles. Bill and I had lunch after the meeting and I learned a thing or two. First, the notion of an Ambassador is not only an excellent idea, it is unique. Bill's job is to make the scene at Atari, Apple and Commodore User Group meetings, demoing software and answering questions. Broderbund is the only company I know of that does this and more companies ought to.

Another purpose of Bill's visits, and one that we really did not have enough time for, is to solicit new products from software authors. What better place than a User Group meeting to find people who really know there machines and who are writing software for it. Both Bill and Broderbund deserve kudos for the job they are doing. We'll have Bill back again sometime in the near future. Note to Doug Carlston (president of Broderbund): Bill is doing a great job, give him a raise! Note to Bill: Hey, what happened to you Wednesday night?

Item:last. Hey, it's a software/ hardware jungle out there. Let's do it to them before they do it to us. And may your April fool's day be unpredictable.

Arthur Leyenberger,
President, Jersey Atari Computer Group

SOURCES
Hardware-Software-Repair

for your ATARI. Asterisks indicate advertisements elsewhere in this newsletter.

GEMINI
86 Ridgedale Avenue
Cedar Knolls, NJ 07927
(201) 267-0988

Wayne Computer Software
1459 Route 23
Wayne, NJ 07470
(201) 628-7318

Pro*Plus
Mike Yocum
3118 N. Prospect
Peoria, IL 61603

Better Mousetrap Software
Frank and Tom Pazel
14 Whitman Drive
Denville, NJ 07834

Software Spectrum
382 Somerset Street
North Plainfield, NJ 07060
(201) 561-8777
Princeton, NJ 08540

The above listings are free to current advertisers. Others interested in being listed in this column should send business address and telephone number with check for $5 per month, payable to JACG, to advertising manager Joseph Rowland, 429 Washington Street, Hackettstown, NJ 07840.

In This Issue

From the Conn - A. Leyenberger.....2

Solid State Display Screen - F. Pazel.....4

Getting Down To BASICS - R. Kushner.....5

The Literature of Forth - D. Forbes.....6

Cartoon Punchline Contest Winners.....7

Leaf Storm - K. McDonald.....8

Trouble At Mayhem Villa - Gerry Poornell.....10

Mentors.....11

March Meeting Highlights - J. Kennedy.....12

Games Atari Plays - J. Kennedy.....12

Computers In Schools - W. Brooks.....13

ATR Talk - W. Morris.....14

Peeks And Pokes - K. Pietrucha.....14

ATR 8000 - What Is It? - R. Lichtenstein.....19

The Power of Forth - D. Forbes.....20

32 BASIC Programs For The Atari - J. Kennedy.....21

Byte-Sized Programming - T. Pazel.....22

Easter Egg.....22

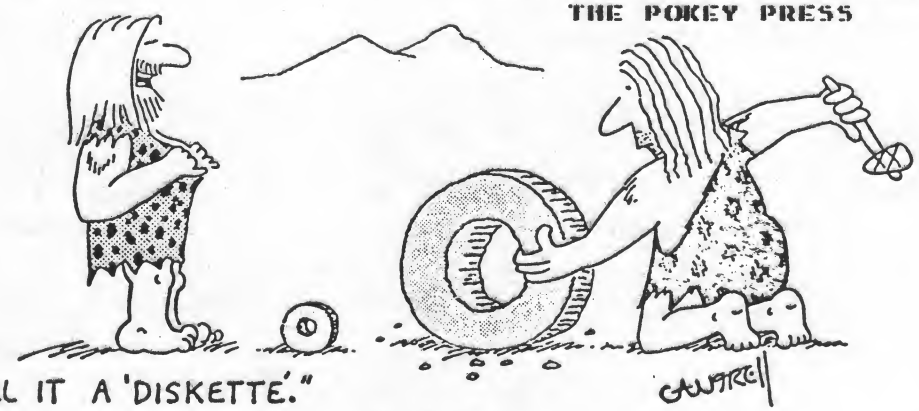
Random Numbers - H. Hirschfeld.....23

Structured Atari BASIC - G. Hampton.....24

To DOS 3 Or Not - T. Marks.....30

Cartoons by Tony Pellechio.....11,19,22

CARTOON BY MARK CANTRELL
THE POKEY PRESS



New Solid State Display Screen!!!

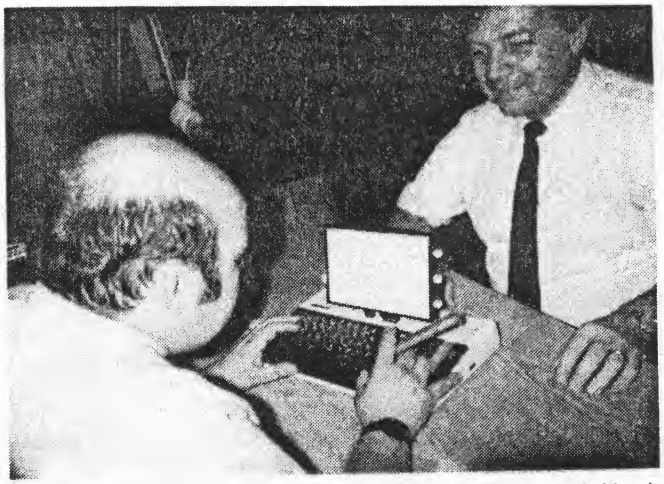
A JACG Exclusive Interview
by Frank Pazel

In a move which caught me totally unprepared a high-level Atari Corporation representative, who must remain anonymous, called on the first Monday of April with an offer that was absolutely shocking. Atari had been having a meeting on Long Island with an EOM of liquid crystal display units regarding a dramatic new product it is going to add to its already impressive line. A second company, located in Bethlehem, Pennsylvania is also vying for Atari's business and was being visited the day of the phone call.

Living in Denville I am only two miles off Interstate 80, the route the Atari entourage was travelling to Bethlehem. Having been in touch with their product development manager lately I apparently lucked out. The representative made me this offer. If I could meet with him he would get off at exit 37 and let me see, try, and even photograph for the JACG Newsletter an exclusive on this new product. In short, I could and he did. Armed with my tape recorder and 35mm camera I waited patiently at our appointed rendezvous, a roadside restaurant. Almost on schedule, the group arrived and we found ourselves a secluded booth in a back corner.

The device will knock your pants off. It has to, without a doubt, set the PC market whirling even more than the announcements made at the recent CES show. The hot new item is officially called the LCDS-2000, standing for the Liquid Crystal Display Screen (2000 referring to the new century). It is exactly what its name implies. Designed for use in the XL, XE, and ST series machines it is a fantastic product. Measuring 15 by 90 centimeters (appproximately 6 by 9 inches) and only about 3/4ths of an inch thick it produces a COLOR display that has to be seen to be believed. As you can see from the photos it simply plugs into the cartridge port on the XL. Power is taken via the cartridge linkage so no external power is required. It weighs an amazing 5.3 ounces which now brings the Atari machines into the realm of being truly portable. With the advent of this invention the proposed XEP (portable XE) machine is definitely on hold.

The picture it produces in full ambient light is more than awesome! Resolution is 1280 x 400 pixels, twice as great as the ST machines are going to be capable of generating. This, incidentally, is while supporting 8 levels each of red, green, and blue. The rep had a modified XL which contained some sample software chips. A demo of a trip up and down a roller coaster was realistic enough to make you feel queasy. The right side of the screen has three knobs; color intensity, tint, and brightness. Even in a well-lighted room the clarity equals that of my 1702 monitor. An additional feature allows the screen to pivot on the cartridge mount so that the viewing angle can be adjusted to suit the individual.



Newsletter Editor Frank Pazel looks on as un-named Atari executive (you might recognize him) demonstrates the new Liquid Crystal Display Screen 2000.

A toggle switch on the left side of the unit allows you to change from 40 to 80 column display. As if I hadn't had enough excitement for one day I was treated to the latest version of Atariwriter in true 80 columns. My fondest computer fantasy had come true.

The bottom line: when and how much? If either of these east coast companies meets the price range Atari wants (they both can already produce in enormous quantities) we can expect them in the stores by July 1. If not, Atari will have to renegotiate with a number of west coast firms whose bids are considerably higher. With the worst possible scenario we are promised appearances in stores by August 15th. This, I am assured, is even with normal slippage. Are you sitting down? The absolutely top retail price will be $49.95! This is in keeping with the new Atari's policy to keep quality up and prices down. They apparently really mean it when they say, "Atari Power Without The Price."

To say that I've barely caught my breath since this experience is an understatement. To also have been given this exclusive (which should break in the press April 15th) is a feather in the cap for user groups at large. That, in fact, is why we were given this information in advance. It would seem that the impact of user groups is finally being realized and this is more than a token move by the corporation to show their confidence in us as the future wave of technical support for the new Atari. In any event, this is a great innovation. Watch out for it.

.....

BOUNTY BOB CONTEST WINNERS

0

Getting Down to BASICs
by Richard Kushner-JACG

This month we will get a chance to see some Quickies. These are short programs and ideas that you may find useful in your own programming. I have collected these items from various Atari groups around the country and have given credit when I knew who to give it to. If you find your own work in this column, with or without credit, thanks for sharing it with us.

QUICKY #1
A graphics demo of interest

```
0 GRAPHICS 24:SETCOLOR 2,0,0:COLOR1
20 FOR X=0 TO 5517
30 SQ=X*X*2.0E-03
40 XCOORD=INT(SQ/191)
50 YCOORD=SQ-XCOORD*191
60 PLOT XCOORD, YCOORD
70 NEXT X
80 GOTO 80
```

Takes time, so be patient!

For those who lost or don't have the AUTORUN.SYS file that is the RS232 handler, here is a BASIC program that creates it. Note that the name it will have is RS232850.SYS. You can, of course, rename it to be AUTORUN.SYS.

QUICKY #2

```
20 OPEN #1,8,0,"D:RS232850.SYS"
30 FOR I=1 TO 88
40 READ D
50 PUT D#1,D
52 ? D;" ";
60 NEXT I
70 CLOSE #1
80 END
100 DATA 255, 255, 0, 56, 75, 56, 169, 80
110 DATA 141, 0, 3, 169, 1, 141, 1, 3, 169
120 DATA 63, 141, 2, 3, 169, 64, 141, 3, 3
130 DATA 169, 5, 141, 6, 3, 141, 5, 3, 169
140 DATA 0, 141, 4, 3, 141, 9, 3, 141, 10
150 DATA 3, 141, 11, 3, 169, 12, 141, 8, 3
160 DATA 32, 89, 228, 16, 1, 96, 162, 11
170 DATA 189, 0, 5, 157, 0, 3, 202, 16, 247
180 DATA 32, 89, 228, 48, 6, 32, 6, 5, 108
190 DATA 12, 0, 96
200 DATA 226, 2, 227, 2, 0, 56
```

QUICKY #3
To disable BASIC, so you don't have to hold down the OPTION key on your Atari 800XL:

-FOR ATARI DOS 2.0S
1. Boot DOS while holding the OPTION button down.
2. Put a diskette containing DOS and AUTORUN.SYS in the drive.
3. Type the following commands:
   a. E (to rename the file)
      AUTORUN.SYS,AUTORUN.OLD
   b. K (binary save file)
      AUTORUN.SYS, D301,D301
   c. C (copy the file)
      AUTORUN.OLD,AUTORUN.SYS/A

-FOR OS/A+ OR DOS XL
1. AND 2. SAME AS ABOVE
3. Type the following commands:
   a. RENAME AUTORUN.SYS AUTORUN.OLD
   b.SAVE AUTORUN.SYS D301 D301
   c. COPY - AF AUTORUN.OLD AUTORUN.SYS

QUICKY #4
When you use the INPUT statement in a BASIC program, the computer includes a question mark on the screen. There may be times when you would rather not have the question mark present, since you can then tailor the display more to your liking. The question mark comes because you are automatically using Channel #0 for an INPUT statement. You can get around this by using "INPUT #16", rather than "INPUT". This prevents the appearance of a question mark, and does not require you to have used any OPEN statement to set up this "Channel #16". Why? Well, BASIC checks the channel number. If there is none, then Channel #0 is assumed. Next, BASIC multiplies the channel number by 16, discards multiples of 256 and checks for a result less than 128 (to prevent the use of channels 8 to 15). In the case of 16, BASIC gets 16*16=256 and subtracting 256 gives 0. Since the result is 0 no new channel is opened! But since it initially found a value other than 0, there is also no question mark prompt! This is an interesting and potentially useful bit of information from the Starfleet Atari (Colorado) group.

We'll continue with more Quickies next month. Or else we'll take up another topic. That's the nice thing about this column. It sort of goes where the wind blows, hopefully providing a warm breeze for those who care to follow it. Bye for now.

THE LITERATURE OF FORTH
PART THREE

by Donald Forbes - JACG

"What is the use of a book," thought Alice, "without pictures or conversations?"

Fortunately, we do not live in the Wonderland of Lewis Carroll. Here are some of the uses of the books from The Forth Source advertised monthly in BYTE by Roy Martens ' Mountain View Press.

METAFORTH, A Metacompiler for fig-FORTH by John J. Cassady is a 80-page typewritten booklet aimed at the FORTH programmer who wants to market an application. Cassady provides three demonstration packages. The first "compiles FORTH together with any application program at a 'base address' memory location. That is, the object code must reside and run at the location where it is compiled." The second demo "compiles FORTH together with any application program at any compile address for any base address...Both the compile and base address must be specified." The third demo "compiles FORTH together with any application program" as in demo two "except that the object code has no name fields and no link fields. It is 'headless' FORTH. It is 'run only' code. It will not interpret, it will not compile, and it is a challenge to disassemble."

Cassady says the code is not a 'turnkey system' or a 'load and go' program but a 'show and tell' demo, a 'this is how it is done, now do it yourself' package. Contact Cassady if you want a license for commercial use.

THREADED INTERPRETIVE LANGUAGES, Their Design and Implementation, was written by R. G. Loeliger of Logicon in Dayton OH in August 1979. When he wrote the book, he says on page 245, "I am sure that FORTH has fully descriptive documents, but they are not publicly available." With a copy of FORTH, Inc.'s 'Microforth Primer' for the Z80 and a DEC User's Society FORTH manual for the PDP-11 he created in August 1978 his own version of FORTH for the Z80. When he offered to do an article for BYTE magazine they came back with a request for a 200-page book manuscript which led to a 500-page manuscript and a 250-page book. If you ever plan to write your own version of FORTH in six weeks, as Loeliger did, you will find the book of interest.

SYSTEMS GUIDE TO FIG-FORTH was written in November 1980 by Dr. Chen-Hanson Ting, a researcher for Lockheed Missiles and Space Company. "Most of the published materials on FORTH are manuals which teach how to use a particular FORTH implementation on a particular computer," he says. "Very few deal with the inner mechanisms on how the FORTH system operates which is essential to the understanding and effective utilization of the FORTH language. My intention here is to describe how the FORTH system does all these wonderful things no other language can. With a deeper understanding of the inner mechanism, a user can have a better appreciation of many unique features which make FORTH such a powerful programming tool...In this book I will attempt to explain the operation of fig-FORTH system in a systematic fashion." If you care about the internals of FORTH, then this is the one book (202 pages) you will want to own.

INSIDE F83 MANUAL by Dr. Chen-Hanson Ting is a brand new book, so that the review will have to wait for a later issue of the newsletter.

FORTH NOTEBOOK dated September 1983 is yet another work by the prolific Dr. Chen-Hanson Ting that runs to 286 typewritten pages. "My experience," he says, "in the last few years in teaching FORTH to people of different backgrounds was that 'Starting Forth' was quite sufficient as a textbook, introducing people to the basics of FORTH. However, to more experienced programmers, the materials in it are not enough...I was constantly asked to provide real applications besides teaching examples. Another shortcoming is that it does not deal with the internals of FORTH operations at the machine code level. There is a wide gap between 'Starting Forth' and the source code as shown in the fig-FORTH model and Installation Manual."

"There are two areas where FORTH books are of urgent need: one is in presenting program design with examples of moderate complexity, and the other is to explore the FORTH computer in a deeper level, dealing with real CPU's and instruction sets. In this Notebook, I collected many programs which I used for teaching purposes. Many interesting games were translated from BASIC into FORTH...I am very interested in the computerization of the ancient oriental GO game. A few programs in this field are included...A number of programs dealing with different aspects of digital image processing...were included here as working examples. I am greatly indebted to the members of the Taiwan FIG Chapter who encouraged me to put together this Notebook." The word 'thesaurus' is the Latin word for 'treasure' and you will find it all here--page after page of executable code. He has a continuous Fourier transform designed to run faster that the Cooley and Tukey FFT, and even has a poor man's FORTH computer written in BASIC!

INVITATION TO FORTH by Harry Katzan, Jr. is a 232-page book written in January 1981 that was prepared on an Apple computer and printed with a dot-matrix printer without descenders. One learns in first grade that the lower case letters G, J, P, Q, and Y extend below the base line, so it is hard to understand the choice of such an unreadable type face. Katzan was chairman of the department of computer science at Pratt Institute and has written several computer books. He belongs to the select company of writers who write one, and only one, book on FORTH. Starting with chapter three, the book is a workmanlike introduction to the fundamentals of FORTH, with exercises at the end of each chapter. Defining words (CREATE, (BUILDS, DOES>) give FORTH its unique ability to create new structures, and thus expand the language, but they are never even mentioned.

FORTH-83 STANDARD by the FORTH Standards Team (a voluntary membership professional

organization) is an 80-page document with a single purpose: "The purpose of this standard is to allow transportability of FORTH-83 Standard Programs in source form among FORTH-83 Standard Systems. A standard program shall execute equivalently on all standard systems." Under 'tradeoffs' the authors state that "when conflicting choices are made, the following order guides the Standards Team: (1) Functional correctness, (2) Portability, (3) Simplicity, (4) Naming clarity, (5) Generality, (6) Execution speed, (7) Memory compactness, (8) Compilation speed, (9) Historical continuity, (10) Pronounceability, an (11) Teachability." Of the 26 team members, which reads like a "Who's Who of FORTH," sixteen hail from California, one from West Germany, and one from Switzerland. The standard, naturally enough, spells out WHAT the components of FORTH will do, but resolutely avoids saying HOW--that is up to the vendor.

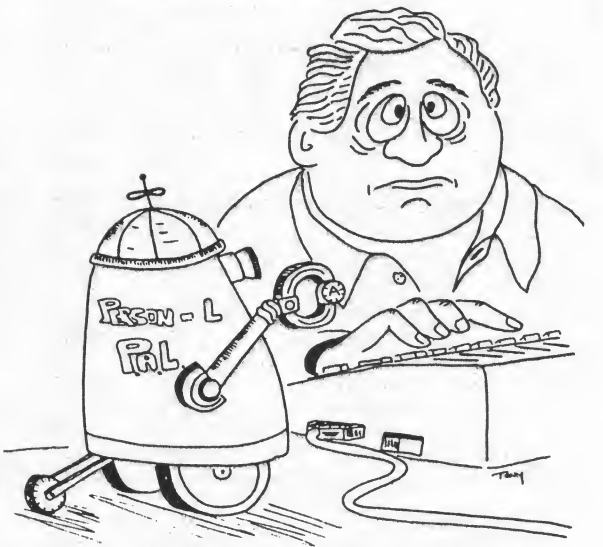
The literature of FORTH has overflowed into magazines such as Forth Dimensions and BYTE and Dr. Dobb's Journal over the past half dozen years. And there is much more to come, as the following extracts from page 371 of the March 85 BYTE indicate:

"Serious FORTH programmers often daydream about...a true stack-based computer whose architecture embodies the FORTH virtual machine. Such a machine has been built by Metaforth Computer Systems of Hull, Yorkshire (England)...It is a single-board processor built in Bipolar logic (with some custom devices), and it uses fast 35-nanosecond HMOS (high-performance metal-oxide semiconductor) RAM chips as its hardware stacks. It has 16-bit memory words and data paths and a 32-bit address bus. The architecture is extremely simple and elegant.
"I've witnessed the prototype running at 2 million FORTH instructions per second with a 10-MHz clock, and the production version...should be capable of at least 5 million instructions per second...The chief designer, Dr. Alan Winfield, has demonstrated that the machine can support a full FORTH-79 system...with a minimum set of 27 instructions, which would make the Berkeley RISC (reduced instruction set computer) look almost baroque...Get more information from Dr. A. Winfield, Metaforth Computer Systems Ltd., Unit 2b, Newlands Centre, Inglemire Lane, Hull, England."
"At this time of writing, NOVIX Corporation of Los Gatos, California, and Charles Moore, the inventor of FORTH, were developing a FORTH microprocessor on a single chip."

HAVE YOU RENEWED YOUR MEMBERSHIP?

CHECK YOUR MAILING LABEL
FOR MEMBERSHIP EXPIRATION DATE
****

CARTOON PUNCH LINE CONTEST WINNERS!



Original cartoon by Tony Pellechio - JACG

Once again, our annual cartoon punch line contest produced a flood of entry. Well, we got 6 lines from 3 people. That ain't bad. Our winners are: Helene Rotundo of Chatham, Bill Brandt of Westfield, and Joe Kennedy of Clark. Here are their side-splitters.

Helene:

ASPIRINS WON'T HELP - IT LOOKS TERMINAL...

Bill:

BUT YOU DID REQUEST ME TO REDUCE THE SIZE OF YOUR BASIC A+ PROGRAM.

BUT YOU DID ENTER THE DOS A COMMAND.

WAS THAT VITAMIN STORAGE OR VIRTUAL STORAGE YOU REQUESTED?

TAKE TWO OF THESE AND CALL THE DISK DOCTOR IN THE MORNING.

Joe:

THE ONLY WAY TO DO IT FASTER IS WITH THE ACCELERATION KEY.

Thanks to all three JACGers for their sense of humor sharing. Okay guys, pick up your free JACG Disk Library offering at your convenience. Just tell Dick Lamb or Dennis Hoskins you won and they will laugh at you.

See there now. Aren't the rest of you sorry that you didn't participate? You could have been a winner too. Promise yourself that when the next golden opportunity like this comes around you will quickly avail yourself of a spot in the winner's circle. Being sorry doesn't help.

LEAF STORM

by Kirk McDonald - JACG

An exciting challenge for the computer graphics enthusiast is the creation of images which are "life-like." The difficulty is that a computer is most easily programmed to produce pictures which are either highly ordered, or completely random. But life is found in a fertile realm somewhere between the crystals and the clouds, and not necessarily in computer code.

One approach to life-like imagery is the use of a graphics input device such as the Koala Pad. In this case the computer merely digitzes a picture created by a human being, and plays almost no direct role in the creative process.

A more ambitious goal is a computer program which draws life-like pictures on the basis of a very small number of initial parameters specified by the user. An important step in this direction has been made by Mehrdad Shahshahani of Boeing Aerospace Co.

[A brief report on his work appeared on page 494 of the August 3, 1984 issue of Science magazine. A highly mathematical discussion of the basis of the algorithm has been given by Persi Diaconis and Shahshahani in Technical Report No. 228 of the Stanford University Department of Statistics, dated November 1984.]

You can explore the strange and wonderful insight of Shahshahani on your Atari 800 or 800XL computer with the aid of the program LEAF STORM. (The name is borrowed from a short story by Garcia Marquez.) This is available on a JACG library diskette in two forms. The source code is in file LEAF.ACT and is written in the Action! language. The binary file LEAF.COM can be run directly from DOS if you do not have an Action! cartridge. The file LEAF.DOC is a text file which contains some instructions on running the program if you do not find it self-explanatory. This file also contains a short description of the mathematics of the program.

To generate a picture you must provide the computer with a list of "transformations." Each transformation is just a set of four numbers. If you specify only one transformation the computer will plot exaclty one dot. But already if you enter two transformations (8 numbers) the computer can draw quite interesting shapes. Figure 1 was produced by entering

```
50 -40 -30 0
50  20 -30 0
```

To my forgiving eye this looks something like the outline of a clump of trees, or a cloud-bank. Of course, we must heed Shakespeare's remark that a cloud may appear simultaneously as a camel, a weasel and a whale to the impressionable.

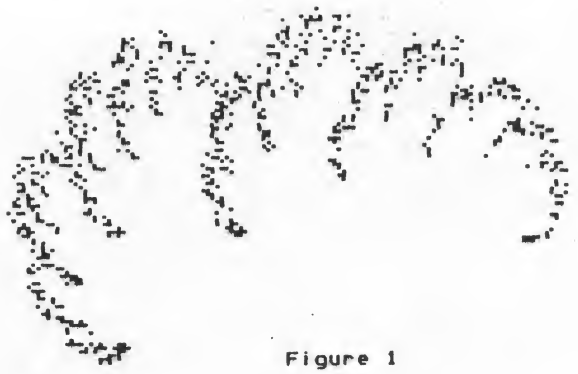


Figure 1

Figure 2 was drawn with the 3 transformations

```
58 30   0 -30
58 70  30   0
58 10 -30   0
```

Figure 2

Figure 3

Figure 3 is an example of the eponymous leaf. Its transformations are

```
50 -43 -31 0
50  46 -29 0
50 -20  -5 0
50  22  -5 0
```

Figure 4 was generated by Shahshahani and is called a poplar treee by him.

Figure 4

These figures do not show a curious aspect of their generation, which would be apparent if you watch them being drawn on your TV screen. The computer very quickly draws a rough outline of the image, and then returns to fill in with greater detail. If the TV screen had infinite resolution, and the computer was left the draw forever, it would add ever finer shading to the picture, without altering the overall form to any great extent. This behavior is closely related to fractals, which are known to be associated with interesting graphics. To my taste, most fractal pictures are rather crystalline, while Shahshahani has succeeded in combining crystals and clouds in subtle proportion so as to simulate life.

Figure 5 illustrates the fractal aspects of the LEAF STORM. The shrimp-like image is generated by the 2 transformations

```
55  10  34  34
55 -34 -21 -33
```

Figure 5

The LEAF STORM algorithm may give us a glimpse as to how complex forms of life can be compactly encoded in DNA molecules. As the pictures are drawn by the computer one gets the feeling of witnessing an evolution from a primitive form to a more and more refined structure. Furthermore, a small "mutation" of the transformation parameters leads to slightly altered images. However, the alteration is not localized to a small region of the image, but is distributed over the whole in a complex way. This simulates the manner in which various members of the same species have a common overall form but differ in detail.

The mutation of the transformations can be readily applied to generate computer animation. Figure 6 shows a somewhat stylized man (Kachina doll?) taking a step. The starting set of transformations is

```
30  10  60 0
30 -10  60 0
50  30 -10 0
50 -30 -10 0
```

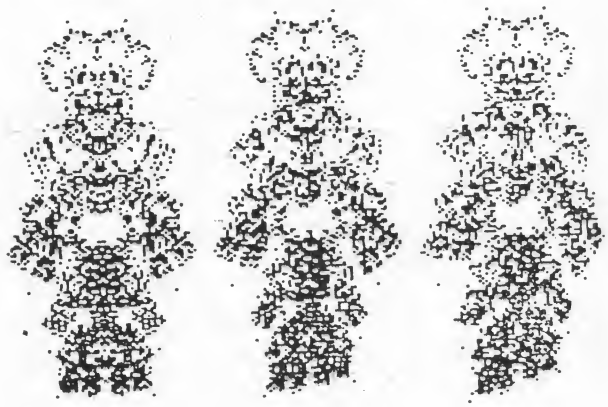


Figure 6

The 2nd step was obtained by changing the two values of 30 to 31 and 29, while for the 3rd step the values were 32 and 28.

Perhaps Walt Disney Studios is in no immediate danger of being replaced by an Atari 800 computer. But when the 520ST becomes available.....

IT'S TIME TO GIVE A BIT

TROUBLE AT MAYHEM VILLA

BY GERRY POORNELL

Welcome to Mayhem Villa for another installment of that never ending investigation of how I can get my hands on free hardware, software and, at the very same time, simultaneously write four fiction books and one encyclopedia all with a simple little word processor that I'm hyping like hell in the hopes of getting as rich selling software as that kid did with Choplifter.

It has been a very exciting month here at the Villa. Our old standby computer, Fido, went to the dogs. It was really something to see. One minute I'm typing away at my usual 5 words per minute speed, and the next minute the screen goes blank and a fieldish voice, speaking through my Blauplunkt amplifier and Bose speakers (which happen to be hooked through my Sony compact disk to the computer) says, " Ha, ha! I have your words trapped inside the computer and you can't get them out." Needless to say, I was upset. This was the last chapter of the second novel, which would also serve as the section from LAR to LEY in the encyclopedia. What was I to do?

Of course, I thought, I must call Zeke at once! Good old Zeke Bleispiel is the only one who knows how to handle Fido, having raised it from a pup. It was Zeke who installed Fido's TAIL (Total-Algorithmic Intelligence Loop) which made it possible for Fido to communicate with his Master Disk. That is, I mean "its" Master Disk. We do have a way of anthropomorphisizing around here, don't we? Anyway, Zeke came right over and, can you believe it, found the problem with only one call to Will Demonmatch in California. The problem was with Will's WD-40 board, which Fido had carried faithfully for two years now. The greebesnap module had fragulated, causing the pull-up transmotor to go into fibrilation. As my old pal Marvin used to say, "They sure as hell don't make 'em like they used to." Marvin was fond of saying that, and he was usually right, having spent four summers after high school with the Mescalera Indian Wild West Show, where he got a degree in Philosophy and Nuclear Engineering. I sure do miss that old varmint. And I sure do miss my Westinghouse no-frost refrigerator with 14 cubic feet of storage capacity and a separate temperature control for the vegetable drawer (not to mention a built-in egg poacher).

As you can see, the month got off on a downer. However, the next day I flew out to Fremont, Nebraska for the annual convention of Word Processor and Fruit Cake Lovers. My son, Ivan, brought along his terrific carrot cake recipe and I brought along Margie, our portable computer with lint remover attachment. As Ivan drove our borrowed Dodge Caravan with 2.2 liter engine, air conditioning, electric door locks and heavy duty suspension, I was able to plug old Margie into the cigarette lighter and write several paragraphs for a new article tenatively titled "Is Divorce Legal on Ring World?".

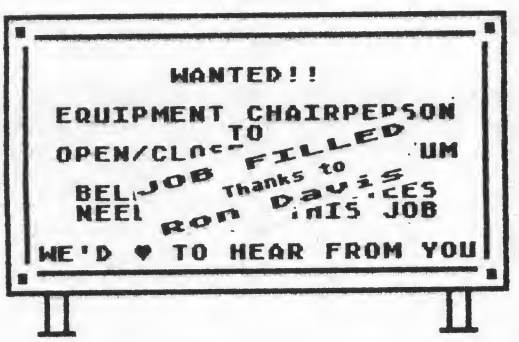
The convention itself was, I'm sad to report, a disaster. Stephan Jokes, President of Lemon Computer Company, failed to show up to deliver the keynote address. It seems he was caught trying to transport an Atari ST computer across a state line. You recall, don't you, that the ST has been banned in 42 states due to its low price and high quality, thus putting the rest of the computer industry out to pasture. I, of course, got one of the first models and I'm sure you read my review of three months ago, in which I said, "Watch out IBM, Atari is about to make computing an essential part of every household in America. At the cost of less than a loaf of bread, the ST computer is the best thing since sliced bread." (I really can turn a phrase, can't I?) And was I ever frosted when the Medford Daily News reported that right before that review appeared in print, a Ferrari appeared at my door, with the license plate that had on it "IM4 ATARI". Pure coincidence, I say. And so does Ivan, who worked his way through college making Pascal flavored ice cream in the electrical engineering lab.

And would you believe what happened when we arrived back at Mayhem Villa? There piled in front of the Andersen triple insulated windows was a bunch of boxes containing all manner of software and hardware for us to try out, review, copy and sell to our friends and then give away as a tax deduction. Let me tell you, it felt like Christmas in March! We haven't had time to even unpack all the boxes as I write this column, let alone review the products. But there is one that has caught my eye and I think is sure to be a best seller. There's this game written by Art Poornell (my brother, but that doesn't influence my opinion in the least). It is the best thing I have ever seen for a micro. The colors are awesome. (I'm talking 5,438,120 colors on the screen at one time!). The sound is incredible (Would you believe the

Mormon Tabernacle Choir in a synthesizer!) The graphics put Star Wars to shame. I managed to traverse only the first level, which by the way is 2,345 screens wide and $1X10^5$ screens high. Even so, I saw a three dimensional map of Disney World, the formation of the Earth from cometary fragments and a visit to the Alpha Centauri as seen through the eyes of a quark. I urge you not to miss this adventure, aptly named "Planet Missionary".

Until next month, when Jerry Pournelle again takes over this column during my extended trip to the Los Rios Rehabilitation Center and Nudist Colony, remember my motto. "Software, Hardware, Underware. Who cares! If it makes money, do it." So long for now from Mayhem Villa.

THIS ARTICLE ARRIVED IN THE EDITOR'S MAILBOX WITH NO RETURN ADDRESS AND A NOTE STATING THAT IF JERRY POURNELLE OF BYTE MAGAZINE HAS ANY SENSE OF HUMOR, HE'LL TAKE THIS ARTICLE IN THE WAY IT WAS INTENDED. IT WAS POSTMARKED APRIL 1.



MENTORS

The following people have volunteered to serve as advisors to our members in the areas specified. Note that some mentors request that you only write to them with your requests. If you can spend some time to help fellow Atarians please contact the Editor.

DATABASES:

Hank Hirschfeld 201/767-1344

TECHNICAL:

Chris Ahlers 201/227-1634

ACTION:

Kirk McDonald 609/924-6167

BASIC or ASSEMBLY PROGRAMMING:

Marvin Kiss
15 Pitman Place
Wayne, NJ 07470

Thanks, in advance, to these volunteers. As the months go by we hope to see this list grow in variety and numbers.

GIVE A BIT!!!

Contribute to the Newsletter this month.

JACG Membership

The Jersey Atari Computer Group (JACG) invites you to become a member. Dues are $20.00 per year and entitle the member to: 1) Receive the monthly newsletter; 2) Purchase programs from the group's extensive tape and disk libraries at special rates; 3) Join special interest groups or form new ones; 4) Benefit from the expertise and experience of other Atari computer users; 5) Participate in group purchases of software at substantially reduced prices; 6) Receive a membership card that entites the member todiscounts at local computer stores; 7) Attend monthly meetings to learn about the latest hardware and software, rumors, and techniques for getting the most out of your Atari computer; 8) Submit articles and programs to the newsletter and give demos and presentations at the monthly meetings; 9) Participate in sale/swap activities with other members; 10) Access the JACG nationally famous Bulletin Board; and 11) Have a lot of fun.

If all of this sounds good to you send a check or money order, payable to JACG, to:

Ron Kordos
201 Lake Valley Road
Morristown, NJ 07960

Remember, receiving the JACG Newsletter is just one of the many benefits of being a member of JACG.

MARCH MEETING HIGHLIGHTS

Reported by
Joseph S. Kennedy

The meeting was preceded by a lively question and answer period which is fast becoming one of the most popular features. Scott Brause answered several questions about the bulletin board. He also reported that the BBS will have Compuserve type menus; the download section will be for JACG members only and that newsletter files can be uploaded to the BBS. Frank Pazel has a bubble sort for DMS.DB. The MOM users group bulletin board is 201-738-4982.

Dick Kushner led a discussion on the lack of member support in general. One outcome of this discussion was that arrangements will be made for a tutorial on any of a wide variety of subjects before each meeting. Art stated that he will again set-up the Hotline since there did seem to be interest from the floor. The by-laws were approved as presented in the February issue of the newsletter.

Frank Pazel pointed out the two contests in the current newsletter; introduced the regular contributors to the newsletter and refused to do a full gainer from the scaffolding on the stage.

Jerry Frese is arranging a repeat of last year's successful Atari Safari and he is looking for volunteers to demonstrate some aspect of Atari computing at the Safari.

Kirk McDonald demoed his program CELLULAR AUTOMATA which he has written in the Action language. This program builds structures of almost lifelike dimensions from seeded input. This program makes math look spectacular. Kirk further volunteered to answer questions on the Action language.

SPY HUNTER from Sega was demoed by Eric Jacoves. The game based on a James Bond-like chase sequence seems to require that the player have three hands. To overcome this glaring lack of a third arm on his part, Eric built a foot pedal from Radio Shack parts which made the game much more playable.

Kirsten Frese gave us an excellent demonstration of GHOSTBUSTERS from Activision. The game is very similar to the plot of the popular movie. (Sit down Dad. Kirsten did a great job.)

Art brought along a little friend of his - Verbot a smiling, voice-activated robot from Tomy. Along with Verbot Art demoed a program he wrote for controlling Verbot with the Atari.

A special treat was the appearence of Bill Holt the Ambassador from BroderBund Software. Bill demoed SPELUNKER, STEALTH, WHISTLER'S BROTHER and PRINT SHOP and gave out copies of each with one of the more obscure of the JACG's raffle formats. Bill also put forth Broderbund's commitment to continue to support Atari.

So that all can sleep better at night,know that Art got his disk drive back. Now, if he could only get his $20 back.

GIVE A BIT!!
****

GAMES ATARI PLAYS

REVIEWED BY JOSEPH S. KENNEDY

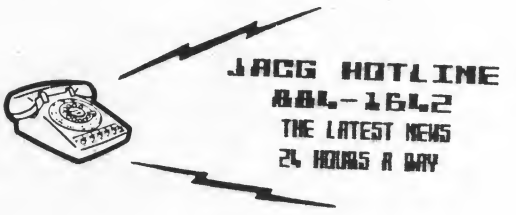
GAMES ATARI PLAYS
BY HAL GLICKSMAN and KENT SIMON
DATAMOST $14.95

GAMES ATARI PLAYS bills itself as the fun way to learn programming. After all, why not program in 29 fun games while you learn to program? However, the fun way to learn to program only works well when the descriptions of the programs explain why certain techniques were used to solve the problem at hand. There is very little explanation of the programs in this book. Further, there are absolutely no views of sample screens from the programs, not even a rough sketch.

I must admit an error above in that there are 29 program listings. But it would take an extremely large rubber band to stretch it to say that there are 29 game programs. Consider that there are such programs included as biorhythm, determining areas, comparing the size of pizzas and an alarm clock. As games go these are right up there with watching grass grow and rust form on bumpers. Probably the two best listings in this book are Land Baron - Monopoly programmed for the Atari - and Cryptogram - a program for helping you work out the puzzles of the same name in the daily paper.

In the "if you can't say something nice don't say anything at all" department it should be noted that this book has two redeeming features. First, it is spiral bound so that it will lie flat while you're trying to type in the programs. Second, the size of the type makes it very easy to see what you're typing.

All-in-all if you want to learn programming from games take the fifteen plus dollars (don't forget the tax) you would have spent on this book and buy three of the disks from the JACG library. You can study the listings of each of the games on the disk as well as having fun playing the games from the very start.



Computers In Schools

by Wm. S. Brooks III - JACG

To me, there is a major problem which faces almost every public school today: how to acquire the money to maintain and enhance the computer hardware and software needed to run an effective program within the constraints of tight budgets? Before I attempt to answer this question you need to know some background information about me and my location.

I work in Massachusetts and am one of two full time mathematics teachers servicing 300 students students in grades 5 through 8. My responsibilities include the teaching of 5 math classes daily as well as all of the computer literacy classes. Along with this computer teaching goes the responsibility for the development, implementation, and evaluation of our computer curriculum and the purchasing, maintenance, and upgrading of our computer facility. Most of the purchases in my area have been made through federal grants or Title IV-B funds, with the maintenance and upgrading of equipment being left to "slush" funds and student, teacher, and/or parent fund raisers. With this in mind, I have found the following suggestions for fund-raising activities and the acquisition of hardware/software very helpful:

1. Make a list of the needed supplies which might be donated by businesses or individuals and circulate it by putting publishing it in the local newspaper and sending it home with the students. I have been able to get all my printer paper and blank disk needs filled this way. Many parents and businesses are very willing to help out a little, and every little bit helps.

2. Contact parents as well as local computer users and let them know what your maintenance needs are. Ascertain if anyone is willing to help.

3. Offer an Introduction to Computers course for parents which might help eliminate their uncertainty of computers and their fear that their children know more than they do. I offer such a course at least once a year, depending on demand, and raise about $300 for each 10 hour course.

4. Run a Computer-Thon, which is like a walk-a-thon, where people get pledges of money for every mile they walk. In this case students get pledges of money for every minute they spend typing in computer programs from computer magazines (like ANTIC, ANALOG, and COMPUTE). One weekend a year (sometime in March to liven up this dismal time) I schedule a Computer-Thon from Friday afternoon until Sunday evening. The weekend is broken into seven 3 hour time slots with no student working more than one time slot. With 13 computers being used I had 91 different students get pledges totaling over $1500. Many teachers, parents, and high school students volunteered their time to help supervise the weekend.

Clearly, this represents a lot of work for the teacher. With the current public image of teachers being what it is coupled with our notoriously low salaries I wonder, sometimes, why we do all of this. The answer, of course, is that we love what we are doing in spite of all the obstacles and we are not afraid to ask for help with what we consider one of the noblest ways to earn a living. If you have budgetary squeezes in your school you might consider doing what we did. And you might find that not only are you bringing in the money that allows you to improve your program- you may discover that it's a lot of fun.

****

HAVE YOU RENEWED YOUR MEMBERSHIP?

CHECK YOUR MAILING LABEL
FOR MEMBERSHIP EXPIRATION DATE

****

ATR Talk

By Walt Morris - JACG

This month I have details of a new peripheral for your ATR 8000- a hard disk drive. Now many of you probably know what a hard disk is, but for those who don't here's a short definition of what one is and what it can do for you.

A hard disk is a device for storing files, just like your floppy disk drives are. Its two major differences are (1) you can't remove the disk and insert another, and (2) the disks can hold much more information because they are sealed, self-contained units. Hard disks (for small personal computers) can hold anywhere from 5 to 150 Megabytes (where a Megabyte is 1024 K bytes). Since floppies can hold a maximum of around 3/4 of a Megabyte (double-sided, double density, 80 tracks), you can see that one hard disk can hold the equivalent of a LOT of floppies. There's also no looking for that one floppy you "just had here last night" or playing "disk jockey" to find a certain file, all the files put on the hard disk are held until erased.

Well, SWP has contracted with another company (COMCPO, 18834 Devonshire, Northridge, CA, 91324) to supply a hard disk for the ATR8000. The disk should be available by the time you read this in April and will hold a total of 8 Megabytes. You will be able to divide this space up as you wish between CP/M files and Atari files, however no MSDOS support for the hard disk will be available (yet). The hard disk package will come with the drive itself, acabinet, a power supply, hook-up cables, and software to format and partition the disk; all for a cost of $895. Also included will be ZCPR3 (See last month's column for a description) for CP/M and the hard disk.

And before you ask, no I don't have one nor do I get a commission on any sales. All the facts in this column were from a phone call made on 3/19, at which time I was told that they intend to have disks ready to ship in two weeks. I was also told they have larger sizes under development, but no estimate of when they might be available.

PROBLEM CORNER

This month we have a not so much a problem as a general warning. A problem arose when a user tried to use a modem program for the Atari to download files and then couldn't get the files off his disk, due to problems with the "file number" (a value put in each sector of a file indicating the position in the disk directory of that file's name).

After talking with the user on the phone, it came up that he was using two different DOSes, SMARTDOS (Rana) and MYDOS (SWP). Although this was not the cause of the problem, it would be to your benefit to make sure that your DOSes are compatible if you want to switch between one and the other and still access all your disk files.

Well, if anyone has any more to say on this topic or problems you need help with, mail them to me at 524 Stratford Rd., Union NJ 07083. Until next month, happy hacking!

PEEKS AND POKES

by Kenneth J. Pietrucha - JACG

As promised, I'm back with the second J.A.C.G. "Peeks and Pokes" column.

One of the most useful locations for me has been location 764, which is known as the 'press any key to continue' location. At times it might be necessary to re-run a program either to make another set of calculations or for some other reason. In this case, I use the following sub-routine (or something similar to it) at the end of my program.

```
100 ?"PRESS ANY KEY TO CONTINUE"
110 IF PEEK(764)=255 THEN 110
120 POKE 764,255
130 GOTO 10:REM GOTO START OF
    PROGRAM
```

The preceeding small sub-routine works like this: if no key is pressed, the value at location 764 is 255, and line 110 is in a continuous loop. The number returned at this location is neither the ATASCII nor the internal code, but is the 'raw' keyboard matrix code. Line 110 will remain in this closed loop as long as the returned value is 255. If any key is pressed, then the value changes. Line 120 pokes a value of 255 back in this location, which makes the computer think no key was pressed, so no key stroke is printed. Line 130 is then executed and the program is re-run.

Another location which goes along with the preceeding discussion is location 752, the 'delete cursor' location. If we use the above routine for 'press any key', no character will be printed on the screen so the cursor is not necessary. To delete the cursor, do a POKE 752,1 and the cursor will vanish from the screen. Any time you want to get the cursor back, do a POKE 752,0.

Here's a novelty location for which I never did find a use. Doing a POKE 755,4 will reverse the letters on the screen. To return the screen back to normal, do a POKE 755,2.

If anyone has an interesting application for this location, I would appreciate hearing from you. To repeat what I said in my first "Peeks and Pokes" column, I would like to focus on applications or ideas for using the locations, as opposed to copying words out of a book and having you figure out your own sub-routines.

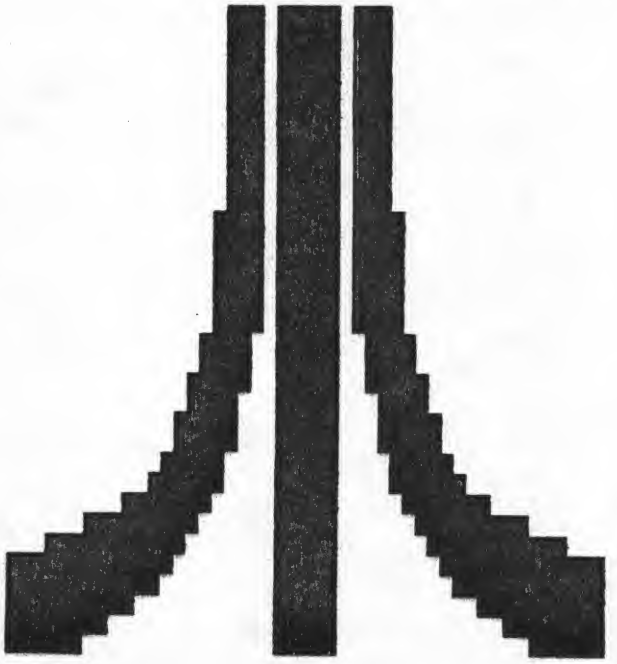
You may have an interesting Peek and Poke application you would like to share with the memebers but don't have the time to write an article. If that's the case, send it to me and let me do the work. My address is 610 Springfield Avenue, Cranford, New Jersey 07016. I'll be looking forward to hearing from you.

This is your newsletter,

Please contribute to it !

QMI

PROUDLY



PRESENTS

TOTALLY

COMPATIABLE
WITH

ALL

400 OR 800
600XL OR 800XL

SOFTWARE

AND

HARDWARE

NEW

XTM201 NON-IMPACT
DOT MATRIX PRINTER
$ 89.95

XTC201 COLOR IMPACT
DOT MATRIX PRINTER
$ 94.95

XMM801 IMPACT
DOT MATRIX PRINTER
$ 159.95

NEW

XDM121 LETT QUALITY
DAISY WHEEL PRINTER
$ 199.95

SM124 HI-RES
GREEN MONITOR
$ 124.95

XC141 14" COMPOSITE
COLOR MONITOR
$ 189.95

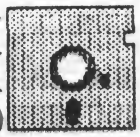
ALL
AVAILABLE
FROM

GEMINI

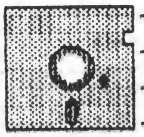
ENTERPRISES

86 RIDGEDALE AVENUE
CEDAR KNOLLS, NJ

267-0988



NEW



ATARI'S

"PROOFREADER"

$ 14.95

ATARIWRITER
SPELLING
CHECKER

NEW

ACCESS THE PLATO
"HOME LINK"
EDUCATIONAL NETWORK

WITH ATARI'S
"LEARNING PHONE"

$ 19.95

(MODEM REQUIRED)
(HOURLY FEE CHARGED)

GEMINI
ENTERPRISES

86 RIDGEDALE AVENUE
CEDAR KNOLLS, NJ

267-0988

STORE HOURS
MON-THU 9:30AM-6PM
FRI 9:30AM-8PM
SAT 9:30AM-4PM

COMING SOON
"ST" SERIES SPECIAL

520ST COMPUTER
WITH 512K RAM
PLUS
SF354 3-1/2" 500K
DISK DRIVE
PLUS
SM124 12" HI-RES
GREEN MONITOR

$ 749.95

ATR 8000
What Is It?

by Robert Lichtenstein - JACG

The March meeting of the JACG was fairly bursting with members and some of the questions raised brought out the need for basic explanations to be provided for some of the newer people. One area of general interest concerns hardware peripherals available for the Atari. One of the numerous "add-ons" is the ATR-8000. This add-on has been spoken about at various JACG meetings and might be one of the least understood hardware interfaces for the Atari.

The ATR-8000 is a co-processor housed in a foot square box which is about three inches high. It is attached to an Atari computer, either an 800 or 800XL, by the cable which usually connects to the disk drive. It can be equipped to operate at three levels of capability (and obviously three levels of price).

At the most basic level, it provides an interface for up to four disk drives (either smart Atari drives or cheaper generic drives), a parallel printer driver with a 4K buffer, and an equivalent RS232 I/O port that supports most of the telecommunications software available for the Atari. By using double-sided generic drives (about $150.00) and MYDOS rather than Atari DOS, it is possible to store up to 360K on one diskette. One can also use Atari DOS 2.0 in the regular mode to boot commercial or other pre-formatted software. One slight problem is the inability to format the flip side of disks in the single-sided mode of MYDOS or Atari DOS, as the ATR-8000 controller uses the timing hole for formatting. However, preformatted flipped disks can be written to and read.

Many generic double sided disk drives have plummeted in price lately and it is quite an experience to have such a large amount of storage available on line. One can even use an 80 track drive which gives 720K of storage on one disk, but these drives are not as compatible with Atari DOS, are more sensitive to the disk media and require special 96 track-per-inch (tpi) diskettes.

At the next level, the ATR-8000 goes beyond the Atari operating system to CPM-80. Within the same box there is a Z80 CPU and 64K of RAM which is accessed with a special program provided by SWP, the builders of the ATR-8000. One can boot CPM on a forty character screen using a color TV, or use a software 80 column screen on a monochrome monitor. The 80 column screen is booted before the CPM system and is almost essential in order to use CPM programs effectively. The 80 column screen is very readable although it doesn't have the "etched" quality of a true hardware 80 column interface. There are literally hundreds of CPM programs available for business, and despite what the "state of the art" is, as reported in many magazines, CPM is still very much alive. In this mode, the user also has a printer buffer of about 48K which provides much quicker system response when printing long files under Atari DOS.

The next level of the ATR-8000 adds an 8088 CPU and 256K of RAM to the above described system. This gives access to MSDOS, and <u>limited</u> IBM compatibility (no it won't run Flight Simulator or Lotus 1-2-3 or any software that uses graphics or video-mapped memory peculiar to the IBM hardware: Video-mapped memory versus RS232 serial display might be a good subject for a future article). This is called the Co-Power 88 upgrade by SWP. One can read MSDOS/PCDOS formatted disks, format them and write to them. Also included with the CO-Power is the ablility to run CPM programs and use the 256K of RAM as an emulated disk drive (Ramdisk). This speeds up programs such as Wordstar, DBase II and other disk intensive programs to almost instantaneous response. It is very useful and actually fun to have this speed, but devastating if the power fails before you've had a chance to save your work to a floppy disk. With the speed, however, saving files often is not a chore at all.

Therefore:

1. The 16K ATR-8000 has the ability to still use all Atari software (copyprotected or not), adds the interface for a printer (with a buffer), modem and up to four much larger capacity disk drives.

2. The 64K ATR-8000 brings 80 columns in software (unfortunately not with Atari programs) and CPM for compatibility with Z80 based computers (some of the software you see advertised in the N.Y.Times on Tuesdays and Sundays).

3. The 256K ATR-8000 adds to this an 8088 processor, MSDOS/PCDOS (more, but by no means all, of those programs from the magazines and newspapers) and the ability to use the 256K as a ramdisk with CPM.

The experience of using the complete system gives a much larger view of the current world of microcomputers than is apparent from using an Atari alone. However, while not terribly expensive for the power available, the ATR is an investment that would have to be weighed in terms of it's utility for each individual.

This certainly doesn't cover everything about the ATR-8000, but gives a beginning view. The ATR-8000 is available with disk drives at Gemini, and support is available from quite a few of your fellow JACG members.

THE POWER OF FORTH

by Donald Forbes - JACG

The power of FORTH lies in its inner interpreter, which acts as FORTH's operating system.

FORTH consists of five parts: (1) the dictionary with its editor, (2) the assembler, (3) the disk that provides a 'virtual' memory extension to the internal storage, (4) two stacks (the data stack and the program stack), and (5) two interpreters, the text (or outer) interpreter and the address (or inner) interpreter.

FORTH is transportable across machines because it is able to put its hooks into a computer and convert it into a FORTH virtual computer. The hooks consist of the machine language interfaces of the inner interpreter. To understand the internal workings of FORTH one needs a grasp of the workings of the inner interpreter.

A description of the inner interpreter can be found in 'All About FORTH' by Dr. Glen B. Haydon, a 278-page annotated glossary of common FORTH ideographs (the words or commands of the language) in the public domain and published by Mountain View Press of Mountain View CA.

Haydon's analysis provides the best explanation to date of the workings of what amounts to the operating system of FORTH. He provides a clear, if technical, explanation of the pointers and registers that interface with the hardware. This, in reality, is what makes FORTH work--all the rest is, in the last analysis, just a form of bookkeeping. Here is the first half of his explanation. (The rest will follow next month.)

THE INNER INTERPRETER

The key to FORTH's power lies in the efficiency of its inner interpreter. However, understanding the operation of the inner interpreter is not necessary for effective use of FORTH. Read this section to better appreciate the power of FORTH.

FORTH provides an interface for software to a wide variety of processors. When a common dialect of FORTH is implemented on different processors, all high level FORTH programs are portable among them. The implementation of FORTH's inner interpreter is processor dependent and will be different for each processor. However, the necessary functions can be described in FORTH.

FORTH uses four pointers. In the implementation of FORTH on a given processor, these pointers may be assigned actual registers or memory addresses. The functions of the inner interpreter must be coded for the particular processor. It is however, possible to make the pointers FORTH variables and to describe the implementation of the functions in high level FORTH. In this manner, FORTH is written in FORTH.

The FORTH implementation described here is known as indirect threaded code. The implementation is different for direct threaded or token threaded FORTH.

The inner interpreter determines the order of execution of FORTH functions. The most common functions are a series of other FORTH functions which are compiled in colon definitions. Such compiled definitions contain a series of addresses to the successive functions to be executed. Colon definitions can nest other colon definitions without limit. Of course the size of memory does impose a limit on the space in the dictionary for new definitions. The inner interpreter is able to find its way through all levels of nesting.

All processors have a 'program counter.' In our virtual FORTH system we make the program counter a variable (PC). Our implementation of FORTH uses two stacks. We use a pointer to the top of each of these stacks: the data stack pointer, (SP), and a return stack pointer, (RP). The data stack is used for temporary storage of data and for operations on data. The return stack is used to keep track of the order of nested functions for the inner interpreter. We make the data stack the same as the processor's hardware stack. We must keep up with two other FORTH pointers: the interpreter pointer, (IP), and the current word pointer, (W). IP points to the next word to be interpreted. W points to the word currently being executed. The key to the function of FORTH lies in the interaction among these pointers.

We define the pointers as variables in our virtual system.

```
VARIABLE  PC  Program counter
VARIABLE  SP  Data Stack Pointer
VARIABLE  RP  Return Stack Pointer
VARIABLE  IP  Interpreter Pointer
VARIABLE  W   Current Word Pointer
```

Ultimately, operation of FORTH is contained in a group of code instructions which are actually implemented in machine language. In our virtual machine we describe the function in high level FORTH.

For a top down description of our inner interpreter we can begin with just colon definitions. The dictionary header of each colon definition includes a code field. The appropriate function is initiated by an indirect jump to a group of machine instructions being pointed to by the code field. This group of machine instructions, in fact all machine instructions, end with a jump to the next group of machine instructions to be executed. The interpreter pointer keeps track of that location. The machine instructions for a colon definition parse a series of addresses contained in the definition's parameter. These addresses point to the successive functions which make up a colon definition. The last function must terminate the colon definition.

For the bottom up description we begin with NEXT which terminates the machine code instructions. It gets the next address value from the IP and moves it to the PC.

This function is factored into two parts:

```
: NEXT1   W @ @ PC ! ;
```

The address pointed to by W is moved to the PC. The new address in the PC becomes the address of the next opcode for the system.

Note: Some implementations of FORTH may increment the address in W at this point; however, it is not necessary.

NEXT then uses NEXT1.

```
: NEXT   IP @ @  W !
      2 IP +!  NEXT1 ;
```

Next fetches the address currently being pointed to by IP and stores it in W. The address in IP is incremented 2 bytes to set it to the next address. Finally, the function of NEXT1 places the address pointed to by W into the system's PC.

The header for each entry in the FORTH dictionary contains a code field. The code field points to a function written in machine code for the system's hardware, whence its name. The code segment pointed to by the code field instructs the program in its handling of the information at the beginning of the parameter field.

The code field for a colon definition points to a machine code segment often given the label DOCOL. This label is not a FORTH ideogram ("word") and it not directly accessible in most FORTH implementationss. DOCOL's function must interpret the series of code field addresses which follow.

DOCOL may be factored. The factor, DOCOL1, illustrates the use of a variable as a stack pointer. Remember, a stack pointer points to the current value on top of a stack. When a value is added to the top of a stack, i.e., pushed, the stack pointer must be moved to the new address before the value is stored there. Also, after a value is removed from a stack, i.e., popped, the stack pointer must be moved to the previous value on the stack.

DOCOL1 saves the contents of IP on the return stack.

```
: DOCOL1  -2 RP +! IP @ RP @ ! ;
```

RP is first decremented 2 bytes, one stack cell, in preparation for pushing a new value. Then the current address of IP is fetched and placed in the new cell on the return stack.

Then the function at DOCOL can be defined.

```
: DOCOL  DOCOL1 W @ 2+ IP ! NEXT ;
```

After storing the current value in IP on the return stack, the value in W is fetched and incremented by 2 to point to the next word to be interpreted. It is then stored in IP and NEXT is executed.

Note: In actual implementation, the value in W may be incremented in two steps which may occur at different points.

A colon definition is terminated by executing EXIT in most FORTH's vocabularies. Rather than continuing through the address cells in the parameter field of a colon definition, the top of the return stack is popped to the IP before NEXT is executed.

```
: EXIT  RP @ @ IP ! 2 RP +! NEXT ;
```

The value currently pointed to by RP is fetched and stored in IP. RP must now be changed to point to the next value on the return stack. Then NEXT is executed. In fig-FORTH, the ideogram, ;S, was used.

(We will continue Haydon's discussion next month, where he describes how the inner interpreter handles variables, constants, user variables, program structures with conditional and unconditional branches, as well as defining words.)

GIVE A BIT!!
****

32 BASIC PROGRAMS FOR THE ATARI

REVIEWED BY JOSEPH S. KENNEDY

32 BASIC PROGRAMS FOR THE ATARI COMPUTER
BY T. RUGG, P. FELDMAN AND T. BARRY
dilithium Press SOFTWARE $19.95

When I first looked at 32 BPFTAC I thought to myself, "Great another book on how to program in Atari BASIC!" (If you believe that I've got some magnificent swamp land you'll probably be interested in.) But this book is not about teaching BASIC but rather using BASIC. Since it is a book for the Atari there are the obligatory game programs. But the authors did not fall prey to the accepted logic that the Atari is only a game machine - program catagories cover applications, education, graphics, and mathematics.

All the programs are well described as to the purpose of the program, how to use it (including actual screen displays - this is something missing from most program listings including those in the major magazines), descriptions of the main routines and variables, easy changes you can make to the program and suggested projects as an outgrowth of the program. These are not the usual "how far did you ride your bicycle in three days" programs. Included are programs for determining your biorhythm, deciding on a choice of action (at a price a lot cheaper than the executive planning programs for the baby HAL's) teaching yourself speed reading or solving the area under a curve.

Those looking to expand their programming abilities will find the descriptions of the main routines and variables most helpful in determining how to put together a program. The suggested project give some interesting ideas from adding certain features to the referenced program to determining what were the values of Abe Lincoln's biorhythm when he gave the Gettysburg Address.

Since this is a computer book it has the usual rather high price for a how to book but this is possibly one that is worth it. One never knows but perhaps while you're learning how to use BASIC you might even learn a little BASIC.

DON'T FORGET!
Contribute an article this month.

Byte-Sized Programming

by Tom Pazel - JACG

I would like to take a little time this month to drift away from advanced programming. Instead, I thought I'd explain something that more than a few people have asked me about at the meetings over the past six or eight months. I'm referring to the question: "I have a BASIC program that displays a high-res picture. How do I save the picture on disk so I can print it?"

Well, the answer is actually painfully simple and straightforward. To some of us, anyway. Even if you know and understand how to do this, it won't hurtcha to read on and refresh your memory (not meant to be punny).

The ATARI (as well as most, if not all, other microcomputers) displays data on the screen from what is stored in RAM. In other words, anything on the screen is also in memory someplace. So, it seems to me, if you know WHAT PART and HOW MUCH of memory is being displayed at any given time, it would be possible to write (save) that memory to disk as a file. Then that file could be loaded into other programs (or straight back into RAM) for any use desired. This is the whole gist of the answer to the proposed question.

The most common GRAPHICS modes involved here are 7+ (Micro-Painter) and 8. I will answer the second part (HOW MUCH memory) first. This is easy. It is a known (now) fact that both GRAPHICS 7+ and 8+16 have 192 mode lines on the screen; each mode line displays 40 bytes of RAM. Therefore, these GRAPHICS modes use 192*40=7680 bytes of memory for the screen data. Other GRAPHICS modes can be computed similarly.

With that done, all that is necessary is to know WHAT PART of memory to write to disk. This is almost as easy. Every time you (actually, the computer) issues a GRAPHICS command, there is a two-byte pointer set up in RAM to tell you the address of the beginning of screen memory. This pointer is placed in locations 88 and 89 (decimal). I hope some of you have a good idea of what's coming.

The following program determines the address of the beginning of screen memory and writes 7680 bytes, starting from that address, to disk:

```
 10 GRAPHICS 8+16
 20 COLOR 1
 30 PLOT 0,0
 40 DRAWTO 100,100
 50 DRAWTO 0,100
 60 REM code to draw some more
 70 REM Find screen RAM
 80 SCRN=PEEK(88)+256*PEEK(89)
 90 REM Write 7680 bytes out
100 OPEN #1,8,0,"D:MYPIC"
110 FOR I=SCRN TO SCRN+7679
120 PUT #1,PEEK(I)
130 NEXT I
140 CLOSE #1
150 REM There's your file!
```

If you type this in and RUN it, you should get a file on disk that is 62 (single density) sectors long. You'll also note that this is the same size as all Micro-Painter files. You can load the picture just created into Micro-Painter and do with it whatever your heart desires (the Delete option probably dominates your thoughts at the moment...).

Of the above code, line 120 might be the only "iffy" one in your mind. If you look at it long enough (2 or 3 seconds maybe), you'll see that what it's doing is writing one (1) byte (PUT) to the disk file (#1) from memory location I. In simpler terms, the FOR-NEXT loop writes memory one byte at a time starting at the beginning of screen memory for 7680 bytes. I don't think it can be much simpler than that.

As an exercise, you might want to try to write the code that would load the file just created back into memory and restore it to the screen. As I've said before, experiment. There is no better way for you to learn more about your ATARI computer. Until next time, may your bugs be little ones.

EASTER EGG

From ACE (NSW), Australia

For owners of Preppie! and Preppie!2. Press these key combinations (before the start of each game) for interesting results:

[SHIFT-CONTROL-M] =TOGGLE MUSIC ON/OFF
[SHIFT-CONTROL-ATARI] = SECRET START LEVEL
[SHIFT-CONTROL-INSERT]= TOGGLE BETWEEN 3-5 LIVES

IT'S ABOUT TIME...

THAT YOU WROTE AN ARTICLE FOR THE NEWSLETTER

R.A.N.D.O.M. N.U.M.B.E.R.S.

by
Hank Hirschfeld-JACG
(201)767-1344

P*E*R*F*E*C*T*I*N*G DATA PERFECT

Three things I would like to cover in this issue are:bugs in Letter Perfect when you get the urge to merge, handy utility and US Doubler continued.

I guess last things first, the US doubler project. As I have reported in the last few columns I had installed the US Doubler in one of my drives and was very happy with the results when used with Data Perfect. I have installed another Doubler in a second 1050 drive and all appears to work fine. I found out later that LJK is now using this at the facility in St.Louis and they indicated that they also were very satisfied.

Below you will find a copy of a basic program from Nora Draper of DAL-ACE. This basic program will convert a Data Perfect database into a standard Atari(2.0)DOS format. This will allow you to use your Data Perfect files with other standard format files. The copy below is for two drive systems. The output file consists of fields of data in the length specified when done with Data Perfect.

Some versions of Letter Perfect Version 6.0 may not work properly when used with the Database merge option. This is shown in two ways,first it will not access the second drive in a two drive system but you can continue by using the first drive; next it will not work in double density mode. Contact LJK if you require an updated version.

PS...It appears that LJK has one of the new 130XE and is prepared to support it with Data Perfect if and when the 130XE ever shows its lovely I/O ports. This together with the new 500K drives would make a super system with Data Perfect.

```
10 REM CONVERT DATA PERFECT DISK
20 REM ATARI DOS FORMAT
30 REM by Nora Draper
40 REM (TWO DRIVE VERSION)
50 DIM T$(128)
60 ? "INSERT DATA PERFECT DISK IN DRIVE 1
     AND PRESS RETURN.":INPUT T$
70 ? "INSERT FORMATED DISK IN OTHER DRIVE
     AND TYPE IN FILE SPEC.(D?:NAME).PRESS
     RETURN.":INPUT T$
80 OPEN #2,8,0,T$
90 I=1
100 READ A
110 IF A=999 THEN 140
120 POKE 1535+I,A
130 I=I+1:GOTO 100
140 S=32
150 POKE 769,1:REM SET TO DRIVE 1 INPUT
160 Z=USR(1536,S,ADR(T$))
170 FOR I=0 TO 127
180 A=PEEK(ADR(T$)+I)
190 IF A=0 THEN 270
200 IF A=2 THEN 240
210 IF A=3 THEN 240
220 PUT #2,A
230 ? CHR$(A);
240 NEXT I
250 S=S+1
260 GOTO 150
270 CLOSE #2
275 ? "CONVERSION COMPLETE"
280 END
290 DATA 104,104,141,11,3,104,141,10,3,104,
   141,5,3,104,141,4,3,169,82,141,2,3,16
   9,64,141,3,3,169,128,141,8,3,169
300 DATA 0,141,9,3,32,89,228,96
310 DATA 999
```

Membership Renewal

Take a moment and look at your mailing label on a recent issue of the JACG newsletter. Check the bottom right hand corner following "Last Issue:". This is the month/year when your membership expires. Try to renew at least one month early. This helps us keep our book keeping in order and avoids your missing any issues of the newsletter.

There are two easy ways to renew:

1. Fill out a membership renewal form in the front lobby before our monthly meeting and present it with $20 (in cash or check) to the Treasurer.

2. Copy the information on your mailing label and send, with $20, to:

Ron Kordos
Treasurer, JACG
201 Lake Valley Road
Morristown, NJ 07960

>>>CHECK YOUR LABEL<<<
>>>TODAY!<<<

Structured Atari BASIC

by Gordon B. Hampton - JACG

This is not a new BASIC for the Atari but a method of writing programs called "structured programming". I will introduce you to structured programming and give some BASIC examples (pun intended) on how to use it on the Atari. I am using Atari BASIC and will give examples in BASIC/XL (from OSS) and Atari Microsoft BASIC as well when they have alternate ways to do the desired task(s). Structured programming can be used in any programming language; I am using BASIC because almost everyone who has an Atari computer, has BASIC on a cartridge or built in. I am assuming some knowledge of computer programming and of BASIC. (I have been programming computers since 1970 and have had my Atari since 2/82.)

First, the reasons you should write structured programs. Structured programs are easier to code because you code small blocks of code at a time, instead of a large block of code (called a program). You can also use "dummy blocks" of code, so you can test your parts of your program before it is completed. Structured programs are easier to follow when reading though them, because each block is small and self-contained (they should be relatively indepentant of other parts of the program). They are easier to maintain (change code for bugs or enhancements) because each particular action is isolated in a small number of lines of code. This goes for the original author who, after a year or more, may of forgotten how he (or she) had written the program. It also helps the person who has to pick up someone else's program, and either debug it or enhance it.

Some definitions (as they pertain to structured programming) are in order. A "program" is a set of code blocks. A "code block" is either a set of code blocks or a program statement, with execution starting at the top (or beginning) and ending at the bottom (or the end). A "program statement" is a line of code that does something in the computer language of choice (like a BASIC PRINT statement).

The idea behind "structured programming" is that you always program in a "top-down" fashion. All "code blocks" should begin at the "top" and end at the "bottom" of the block. Within a particular block, it can "jump around", but should follow certain rules of structure that I will state. The rules are not concrete but rather guidelines on how to code programs.

I stated earlier that a structured program is a set of code blocks. The first block should be the main logic of the program and at the end of the block, the program should stop executing. The rest of the blocks are subroutines executed by the main logic block of the program, or by other subroutines. A sample main logic block is something like this:

```
100 GOSUB 1000:REM INITILIZE ALL VARIABLES

200 GOSUB 2000:REM OPEN ALL FILES

300 GOSUB 5000:REM PROCESS ALL DATA

400 GOSUB 4000:REM CLOSE ALL FILES

500 END:REM THE PROGRAM STOPS
```

Notice that I am using subroutines for all program flow, and I comment each GOSUB. I could put the actual code in the main block, but that would make the main block very long in a long program. A rule of thumb is to try to make each code block small enough to see the whole block at the same time (on the same page of printed output or on the same TV screen). The comments give the reader of the program an idea to what the block is accomblishing. On the above, you know what each routine will accomblish except the one the starts at line 5000 (referanced in line 300). To find out what it does, you would have to go to line 5000, where, ideally, are additional comments explaining what this routine does. I know that comments take up valuble memory. If you find this a problem, I'd suggest you buy (or better yet, write) a program that elimiates comments in programs. The output program can be used as the "execution" copy of the program, but keep a copy of the one with comments around for future updates. You should avoid too many statements on the same line. This makes the program hard to read and hard to maintain. It does cost some memory, but my answer to that is the simular to my answer about comments. Notice that the subroutines are not in the order that they are executed (the subroutine line numbers are not in order). This is allowed (I left the largest block to last) because each block is independant (or self-contained) and their order of placement does not matter. You should not assign variable names to your subroutine line numbers because this makes finding them very hard, the program hard to follow, and if these variables are changed in the program, almost impossible to debug. If you need this kind of logic, try using the ON..GOTO/GOSUB statements instead (explained later).

There are 3 types of code blocks possible. (This means that with a series of code blocks, each of which fall into one of 3 catagories, a program that can accomblish anything that can be done on the computer can be written). Another rule of thumb is, whenever you have a situation which cannot fall into one of these 3 types of code blocks, you should split up the code block up into two or more code blocks that are of these types of blocks.

The simplist code block I call a "simple code block". It contains a series of code blocks that are each executed in order. (Remember a code block can be just a program statement). There are times you need a "null code block" which falls into this

catagory. (A "null" code block is one that does nothing at all and probably contains either no code or just comments). Uses for the null block will be descibed later. The main logic block I listed previously is a simple code block. Here is another example:

```
1000 REM PRINT NAME ROUTINE

1010 GOSUB 1200:GET NAME

1020 PRINT#2;NAME$

1030 RETURN
```

Notice that I didn't comment every line. The PRINT and RETURN statements in this case are self-defining. If the code to "get name" is complicated, it doesn't complicate this code block. That logic is independant of this block. The only requirement that this block has of the subroutine at line 1200 is that the name be placed in a variable called NAME$ (used by line 1020). Now, isn't that simple?

The "conditional code block" contains code that is conditionally executed, in other words sections of the block may or may not execute, depending on some condition. If a certain condition is true one code block is executed but if it is false another code block is executed. Either code block may be a "null" block, when you only need to do something under one condtion (true or false). Atari BASIC has an IF/THEN statement that can be used. Coding a conditional block looks like this:

```
390 REM START OF CONDITIONAL BLOCK

400 IF A=B THEN 450

410 GOSUB 500:REM THE FALSE CONDITION

420 GOTO 460

450 GOSUB 600:REM THE TRUE CONDITION

460 REM END OF CONDITIONAL BLOCK
```

The above code block will work in BASIC/XL and Microsoft BASIC, but both of these have alternatives as well. In BASIC/XL you can code this:

```
390 REM START OF CONDITIONAL BLOCK

400 IF A=B:REM THEN

410   GOSUB 600:REM THE TRUE CONDITION

420 ELSE

450   GOSUB 500:REM THE FALSE CONDITION

460 ENDIF:REM END OF CONDITIONAL BLOCK
```

Notice the missing THEN statement (I commented or REMarked one in). This was required so BASIC/XL can tell the differance between the normal IF/THEN and the special IF/ELSE/ENDIF statements. This keeps BASIC/XL upward compatable from Atari BASIC, but also allows for several statements as part of the condition, without putting too many statements on the same line. In Microsoft BASIC, you can code this:

```
390 REM START OF CONDITIONAL BLOCK

400 IF A=B THEN GOSUB 600 ELSE GOSUB 500

460 REM END OF CONDITIONAL BLOCK
```

This is pretty simple to understand, but again I would caution against putting too many statements on the same line. I like the approach BASIC/XL took better (for thid reason), but the Microsoft approach is more conventional with other computer languages (although other languages ussually let you code IF/THEN/ELSE on several statements). The first approach I listed (the only one that works in Atari BASIC) does the job just fine.

There is also another alternative in all 3 BASICs for a conditional code block. The ON..GOTO/GOSUB statements. They are coded like one of these:

```
400 REM SAMPLE ON..GOTO

410 GOSUB 1000:PRINT MENU AND GET "SELECTION"

420 ON SELECTION GOTO 2100,2200,2300,2400,2500,3000

430 PRINT "Sorry, but a selection of ";SELECTION";" is not allowed"

440 GOTO 410:GET ANOTHER SELECTION


400 REM SAMPLE ON..GOSUB

410 GOSUB 1000:PRINT MENU AND GET "SELECTION"

420 ON SELECTION GOSUB 2100,2200,2300,2400,2500,3000

430 IF SELECTION <> 7 THEN GOTO 410:GET ANOTHER SELECTION

440 END:REM ON 7 THE PROGRAM ENDS
```

These whole code blocks are "looping code blocks" (explained next), but the ON..GOTO/GOSUB on line 420 of each are what I want to explain. They can be catagorized as shortcut methods of writing several conditional code blocks. The same function can be accomblished with many IF statements, but ON..GOTO/GOSUB is shorter and more presice. The ON..GOSUB is more "structured" than the ON..GOTO, because you know from looking at this code that eventually you will return to line 430. On the ON..GOTO, you need to look at each routine to find out for sure what will happen.

The last, and most complicated code block is the "looping code block". This code block starts with a code block. It then has a condition test to see if the loop should be repeated or ended. It then has another code block and then a jump to the beginning of the loop. It ends with an exit. Either code block may be "null", and when the second one is "null", the jump and the exit are usually combined with the condition test. The "exit" can also be a "null" code block that marks the end. Under all conditions, the exit is the place to go when the loop no longer needs to be executed because it is the "bottom". The code block before the test is always executed at least once, but the code block after the test may never be executed, if the condition test so designates. The ON..GOSUB example I gave above is a looping block. The ON..GOSUB statement is essentially a simple code block consisting of a series of conditional code blocks. The IF statement is the condtion test to determine if the loop is repeated. The last code block is "null" (not used), and the jump is combined with the condition test. The END statement is the exit. The looping code block is relatively complex, so I am going to give 3 more examples that will work in all 3 BASICs. First, another one with the test at the beginning (or the first code block is "null"):

```
50 REM GET NAME ROUTINE

60 IF N>MAX THEN 100:REM SHOULD I  EXIT
THE LOOP?

70 GOSUB 200:REM GET BYTE FROM FILE

80 NAME$(LEN(NAME$)+1) = ACHAR$

90 GOTO 60

100 REM END OF THE LOOP
```

Notice the indepenance of how the byte is gotten from the file from this code block. It just doesn't matter here how it is to be done, as long as a character is placed in ACHAR$! Next, let'S look at a block with the test at the end (or the last code block is "null"):

```
50 REM GET NAME ROUTINE

60 GOSUB 200:REM GET BYTE FROM FILE

70 NAME$(LEN(NAME$)+1) = ACHAR$

90 IF N<=MAX THEN 60:REM SHOULD I
REPEAT THE LOOP?

100 REM END OF THE LOOP
```

Notice that in the first loop, the loop might never be executed, but in the second, it must be executed on the first pass. Also, the jump to the beginning of the loop has been incorparated into the condition test. Finally, a looping code block with the test in the middle (or with no null blocks):

```
50 REM GET NAME ROUTINE

55 N=N+1:REM INCREMENT THE POINTER

60 IF N>MAX THEN 100:REM SHOULD I  EXIT
THE LOOP?

70 GOSUB 200:REM GET BYTE FROM FILE

80 NAME$(LEN(NAME$)+1) = ACHAR$

90 GOTO 55

100 RETURN:REM END OF THE LOOP
```

If you had wandered where N was being incremented in the other examples, here it is obvious (line 55). We are assuming N is initilized when our routine is entered. In the other examples, N must of been incremented as part of our "get byte from file" routine, or we would have an endless loop (a loop that never stops). Notice in the last loop that N is always incremented, but the other code is dependant on our test in line 60. The looping code block as I have described it, can handle all possible loops, but there are alternate ways of coding them available. In all three BASICs, there is the FOR/NEXT loop. It looks like this:

```
100 FOR I = 1 TO N

110 PRINT A(I),

120 NEXT I
```

At first glance, this seems to be a "looping code block" with the test at the beginning (or the first code block is null). Well, this is only true in Atari Microsoft BASIC. If N is less than one, Microsoft BASIC will not execute the loop. However, Atari BASIC will <u>always</u> execute the loop at least once. OSS's BASIC/XL acts the same, because of compatablity with Atari BASIC. So, these 2 act like a "looping code block" with the test at the end (or the last code block is null). This differance causes confusion for people trying to convert programs from one BASIC to another. You can avoid the problem altogether if you code the loop like one of the following:

```
100 REM ALWAYS EXECUTE AT LEAST ONCE

105 I = 1:INITILIZE I FOR THE LOOP

110 PRINT A(I),

113 I=I+1

120 IF I <= N THEN 110


100 REM EXECUTE ONLY WHEN NEEDED

105 I = 0:INITILIZE I FOR THE LOOP

110 IF I < N THEN 120
```

```
113 I=I+1

115 PRINT A(I),

117 GOTO 110

120 REM END OF THE LOOP
```

These work the same in all BASICs, and it is clear what they do under all conditions. If you feel unsure of the FOR/NEXT loop, you can code like the examples above and feel safe. At this point, to be complete, I'll mention one more alternative of the "looping code block" only available in OSS's BASIC/XL. It is the WHILE/ENDWHILE loop. I will use as an example how to code the original FOR/LOOP using WHILE/ENDWHILE, but achiving the result of Atari Microsoft BASIC (in other words, the loop does not have to execute):

```
100 I = 1:INITILIZE I FOR THE LOOP

105   WHILE I<=N: REM SHOULD WE  EXECUTE
THE LOOP

110   PRINT A(I),

115   I=I+1

120   ENDWHILE
```

The test is at the start (line 105), and if the condition is not true the first time, execution will jump to the ENDWHILE (line 120).

Now that I have the 3 code block types out of the way, I want to give a tip for those who use (or will use) structured programming. You can code a particular code block as a "dummy" block. For example, if you are going to have the program draw a picture on the screen, but you don't want to code that code block until later, you can code a dummy block instead. In this case, it would be something like this:

```
10000 REM CODE TO DRAW A WORLD MAP

10010 PRINT "MAP  OF THE  WORLD IS  NOW
DRAWN"

10020 RETURN
```

You can put this code in the program, and test the program for everything except the picture. When the picture would normally have been displayed, you get the message instead. Now, whenever you are ready, you can change the code block above to whatever code it would take to get the picture drawn. After that, if the picture is not right, you know where the problem must be, because the rest of the program has been debugged.

Just think, after your program is completely debugged and is working perfect, you can forget about its code. Then, when your "perfect" program stops working properly, you should be able to pinpoint the problem code very quickly. For example, if you get 1E+24 (a whole lot of) messages saying your file has no records, you know to go to either the routine which prints out this error message or the routine that calls it, which is probably the routine that reads the file.

In the examples of code I gave, I almost always let GOSUBs represent a code block within a code block. This does not have to be the case. You can write the particular code block right in place. Just don't let the code block that contains it get so big that it can no longer easily be looked at and understood. But when in doubt, use a subroutine (GOSUB).

I have written programs that were not structured. I am usually sorry when I have to go back to these programs. I have taken programs (usually someone else's), and rearranged code (sometimes significantly), just to make it structured, before having to do some maintenance to it.

Some people swear by flowcharts, but these people do not usually code programs, just teach how to. If you write structured programs, you won't need a flowchart. Each code block is small enough to be self documenting and easy to follow. A flowchart is a very good learning tool.

You can write a program in any programming language that is structured, including assembler. All it takes is a small amount of forethought. The program should end up taking less time to code and debug than it would have if it was coded unstructured because its less complicated. Any program can be written as a "structured program".

"Relax — it's just out of paper."

S.L.O. POKES
San Luis Obispo, CA

BASIC ATARI BASIC

By Richard Kushner - JACG
An evaluation by D. Forbes - JACG

I had a hole in my shoe. My good wife Judy took me to the Livingston Mall to buy another pair. We stopped by Waldenbooks--as always. At the computer book racks, a couple and their son were trading questions. I butted in: "What computer are you interested in?"

"Atari."

"Oh, yes, Atari," I said. "If you are going to buy one book, buy that blue one by Poole that you have in your hand. If you are going to buy two books, you need 'Basic Atari BASIC' by James S. Coan and Dick Kushner that's right here on the shelf. We have 500 people in our user group that meets on the second Saturday of each month in the auditorium at Bell Labs in Murray Hill. You're welcome to come and it won't cost you a penny."

Downstairs we peeked in at B. Daltons. I knew Coan had written 'Basic BASIC' and 'Advanced BASIC' for Hayden Books. He had two more books on the shelves: 'Basic BASIC for the APPLE' and 'Basic BASIC for the COMMODORE 64.' Evidently a prolific writer.

There was a good reason for suggesting Dick Kushner's book. I have a copy ($15) in my hand inscribed: "To Don Forbes - Best of luck with FORTH and BASIC. Richard G. Kushner 11/10/84." I asked him how he came to write the book.

He said that when Art Leyenberger edited the newsletter they had an empty space to fill so, as a joke, they labelled it: "This space left blank on purpose." Another publication thought the joke was funny, so they wrote it up. Dick's name came to the attention of Hayden Books and they asked him to collaborate with James Coan on a book on the Atari. Dick decided that he wanted to do a book that would do justice to the Atari and not be just a translation of a book on BASIC for the Apple or something else.

Dick says he met with Coan just once (Coan lives somewhere in Pennsylvania) but they conferred many times at length over the phone. About a year later the book was finished.

When Dick (who is 44) stepped down as president of the JACG last year, I discovered that he had a doctorate in chemical engineering and was doing fundamental research on advanced semiconductor process control for Bell Labs.

You have to hand it to the guy. You wonder how he managed to run the JACG (including West Coast travel) and write for the newsletter on top of a fulltime job, and then find time to write a book on the side.

I decided to get my money's worth and started copying the BASIC programs from his book on to disk, one by one. The book is clean. You can usually tell a sloppy job by the number of typographical errors and misprints.

The exercise was rewarding. Dick evidently has a good grasp of the inner workings of the Atari and of the many tricks that you can use to get the best use out of the machine. Coan is an experienced writer who spells everything out clearly--you never have to reread the text to find out what he is driving at.

The book is designed for 'learning by doing.' The core of the book is 150 short, complete programs that move forward "building one on the other until you reach your goal."

"We encourage you to experiment," Dick says. "There is no way that you can damage your computer by typing in an incorrect command. The very worst thing that can happen is that, under certain circumstances, the computer will 'lock up'--that is, it will refuse to respond to any key that is pressed (including, sometimes, the system reset key). If this happens to you, you will have to turn the computer off and then back on to regain control. In the process you have lost any program that was in the computer memory. Since most of the programs in this book are very short, you can easily retype the lost information. You will make lots of mistakes and be greeted with many error messages. That's a sure sign that you are learning! It is rare for all but the shortest programs to run as expected the very first time."

Chapter one shows how to enter data and obtain results from the computer. There is a program to calculate gasoline mileage and how to create graph paper on the screen.

Chapter two shows how to plan a program. There is a package weight monitor, and a random number generator that will flip a coin 38 times and roll a die ten times, as well as a program to check for bad input that could crash the system.

Chapter three introduces graphics, and shows how a gambler can draw a die anywhere on the screen. There is a graphics program which combines graphics 3, 5 and 7 in a flashy demo.

Chapter four shows how to calculate compound interest with money added each month, and how to use the paddles and joystick for input. The chapter ends with a program to use the joystick to draw lines on the screen.

Chapter five is devoted to string handling. You can rearrange a string in alphabetic order, and even rearrange 'John Jones' into 'Jones, John.' You can also find the byte representation of any character. The letter B, for example, is composed of 0, 124, 102, 124, 102, 102, 124 and 0. You can then redefine any character in any way you choose, and thus create your own alphabet.

Chapter six covers numeric arrays and how to simulate string arrays. The examples allow you to play a Geography game in which you must guess a name that begins with the same letter as the last letter of the previous place name. You can also make your own clock with the timers that are built into the computer.

Chapter seven shows how to break a number into its digits, how to convert a number from decimal to binary to hexadecimal and back to decimal. You also find out how to compute the number pi (3.1415) by plotting random numbers in a square with an inscribed circle.

Chapter eight proves (if you still need proof) that Dick did a careful job. He devotes twelve pages to the tape recorder for those of us who cannot afford a disk drive, and shows how the geography game can be saved to tape.

Chapter nine discusses input and output using the disk drive, and features random-access files. You learn how to develop a mailing list program, and how to enter names in a mailing list file.

Chapter ten is devoted to sound. "Games demand sound. Crashes and whistles and laser sounds all add to the fun of a game. Music by itself or as an introduction to a program contributes to user enjoyment. Sound effects that indicate user choices are also effective. Beeps indicating that a choice has been made, buzzes for incorrect or illegal choices, musical chords for correct choices--the potential uses of sound go on and on. Sound can be as effective an attention grabber as graphics."

Francois-Frederic Chopin, who died in Paris in 1849, stands in the same relation to piano music that Beethoven does to the symphony, Mozart to the opera, Handel to the oratorio, and Schubert to the 'Lied.' His name will live forever as the author of the song "I'm Always Chasing Rainbows," from the slow, middle theme of the Fantasie-Impromptu in C-Sharp Minor, Opus 66 which was among the manuscripts that his friend Julian Fontana published after his death. Dick devotes page 207 to the music notation for the theme of this immortal melody, and the program on page 204 plays the music.

Chapter ten is the meat of the book--fifty pages devoted to Atari's colorful graphics, which have not yet been surpassed. Even today you cannot find a match for Atari's graphics on the IBM PC, or AT&T's PC, or the Apple, or the Macintosh, or the Commodore 64, or the TRS-80.

In this chapter he covers all the graphics modes, from 0 through 12. He even has a section on the four graphics modes 12 through 15 available on the Atari 1200 and XL models. He takes you through player-missile graphics, collisions with playfields, vertical player movement, multiple screen formats, and how to develop a custom display list.

The book can be used as a teaching text because each chapter ends with a set of exercises, and the book concludes with the answers to the even-numbered problems.

Dick, unlike so many other authors, realizes that just because you are ignorant doesn't mean that you are stupid. So he very considerately concludes each chapter with a Programmer's Corner which highlights special features or advanced programming ideas.

"Programmer's Corner 1 elaborates on the Atari screen editor and the graphics characters. The discussion of the editor continues in Programmer's Corner 2, where error trapping is also covered. Programmer's Corner 3 also extends the discussion of graphics in that chapter and introduces BASIC keyword abbreviations, while Programmer's Corner 4 investigates the attract mode and improved input for programs. Programmer's Corner 5 shows how to check whether a key has been pressed and how to modify the character set. Sorting, making a built-in clock, and using the START, SELECT and OPTION keys are the topics of Programmer's Corner 6.

"A menu program is featured in Programmer's Corner 7. Programmer's Corner 8 continues the theme of that chapter with a discussion about overcoming tape loading problems and automating tape loading. Programmer's Corner 9 presents a detailed description of the disk utility commands, and in Chapter 10 the Programmer's Corner shows how to translate sound into music. The final Programmer's Corner presents a technique for modifying the screen to display more than one graphics mode at a time."

If you have been led so far to believe that this book has everything, then you are sadly mistaken. There are absolutely no cartoons. There is not even a single joke. The biographical information is limited to a laconical New Hope, PA and High Bridge, NJ. And the only first person reference is on page seven where the Atari computer prints: "I don't spell very well."

Writing For The JACG Newsletter

****

Articles should be submitted to the Editor by the 20th of the month for inclusion in the next issue. Submissions preferred on disk, using LJK Letter Perfect or Atari Writer. Font style should be Elite or Proportional with right hand justification. If hard copy is submitted the final printed width should be 4-1/4 inches from left margin to right margin. All formats will be considered including hand written documents if first arranged with the Editor.

We want to encourage everyone to voice his/her thoughts, knowledge, and opinions. Writing will be modified at the discretion of the Editor. No piece will be knowingly altered out of original intent.

****

NEXT MONTH!!
YOUR ARTICLE

Could Be In This Space
And Beyond.....

TO DOS 3 OR NOT, THE NEWCOMERS DILEMMA

by Ted M. Marks - JACG

When I bought the 1050 disk drive in October for my new 800XL, my dealer made sure that I got a recently manufactured unit with the new DOS 3 included. He did not do me a favor. My friends who had a some knowledge of the "new improved" DOS suggested I stick with good old DOS 2.0S, but as usual, I did not listen. The prompts, the default features and the Help screens seemed to make DOS 3 perfect for a computer beginner. Besides, the Access DOS 2 feature allowed "easy" conversion of files. The "dual" density (i.e. 140 % of DOS 2) meant more programs per disk. I am writing this article in order to dissuade anyone just starting out from using DOS 3. There are major flaws in the above logic.

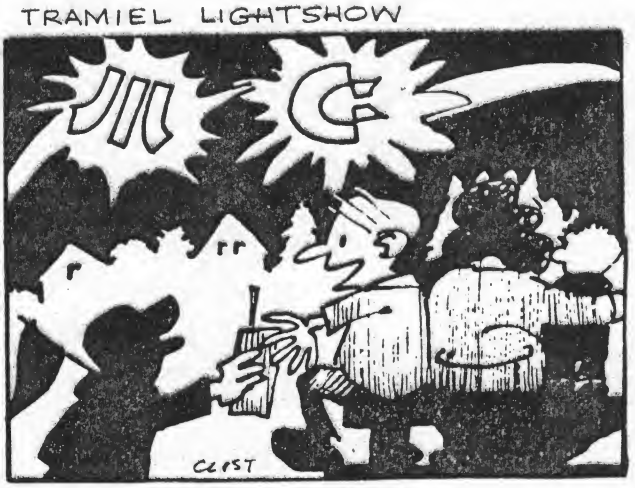
The April 1985 Antic has a fine atricle which describes DOS 2.0S to new 1050 owners. They say that "Antic strongly recommends that all new owners use the earlier DOS 2.0S until they feel comfortably knowledgeable with the DOS functions." They seem to imply that after a short time you should go on to the more advanced DOS 3. After having used DOS 3 rather extensively for a few months for fairly routine file manipulations, my opinion is that a notch be cut into the DOS 3 Master provided with the drive and the disk used for other programs.

There are really two reasons for this view, either one of which would justify never taking the DOS 3 Master Diskette out of the envelope. The first reason is DOS 3 is just not convenient to use. To do the routine keyboard functions of viewing the directory, lock, unlock, rename, binary save or load and delete, you need three files: FMS, KCP and KCPOVER. These should be on all "working disks." These take up 10 blocks or the equivalent of about 80 DOS 2 sectors, which is the same as the DOS and DUP files of DOS 2. To copy files, duplicate a disk, initialize or convert 2 to 3, you need to load a separate file for each function. These files are kept on the master diskette. This means a lot of disk swapping. If you want to remane or protect during a copying session, the copy utility (or any of the others) must be reloaded. Even checking the directory dumps the program.

The second, and equally valid reason is the incompatibility of DOS 3 files to those of the rest of the civilized world. Sure, you can convert your friend's files as well as JACG Library disks or other purchased software to DOS 3, but going the other way is not easy. The January 1985 Antic had an article appropriately titled "Escape From DOS 3," but the procedure seems cumbersome. You could also use a cassette recorder to hold your program while you reboot your computer with DOS 2, but this seems hardly worth the effort.

If these arguments aren't enough to convince anyone not to use DOS 3, there is now a third reason. The XE machines will, according to Antic, come with a new DOS 2.5. It is said to be entirely compatible with DOS 2.0S. This means that the millions of new Atari computer owners will not be directly compatible with DOS 3. Anyone not convinced?

TRAMIEL LIGHTSHOW



"HURRY! THE FIREWORKS IS JUST BEGINNING!"

PORTLAND ATARI CLUB

J A C G
JERSEY ATARI COMPUTER GROUP
40 LAWRENCE ROAD
PARSIPPANY, NEW JERSEY 07054

FIRST CLASS MAIL

[redacted]

JACG NEWSLETTER — VOLUME 4, NUMBER 8

April 1985

J A C G EXECUTIVE COMMITTEE